Crawling in the Deep: Evaluating Web Crawling Configurations for Web Privacy Measurements

Karthik Ramakrishnan
Georgia Institute of Technology
Atlanta, Georgia, USA
rkarthik@gatech.edu

Qinge Xie
Georgia Institute of Technology
Atlanta, Georgia, USA
Zhejiang University
Hangzhou, China
qxie47@gatech.edu

Uma Anand
Georgia Institute of Technology
Atlanta, Georgia, USA
uma@gatech.edu

Frank Li
Georgia Institute of Technology
Atlanta, Georgia, USA
frankli@gatech.edu

Abstract

Web privacy measurements fundamentally depend on web crawling techniques. Unlike many other web measurements that can be performed only using the landing page (e.g., TLS assessments, web server security configurations), web privacy evaluations often must explore deep into a website to more comprehensively identify and characterize its privacy behaviors. However, prior studies adopt various crawling configurations and strategies, and to date, we lack a systematic understanding of the impact of different crawling configurations on the privacy behaviors observed.

In this paper, we address this gap by conducting controlled web crawling experiments across the top 10K CrUX websites, while monitoring privacy-related metrics including HTTP request domains, JavaScript Web APIs accessed, and cookies set. We evaluate 12 different crawling strategies, varying the page search strategy (depth-first, breadth-first, and a hybrid approach), whether browser state is preserved across page visits, and whether user interaction is simulated. We further assess the impact of varying the number of pages crawled per site, as well as the amount of time the crawler waits per page. Our analysis shows how different crawling parameters influence different privacy metrics, ultimately informing recommendations for designing web crawling methods for web privacy measurements.

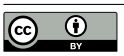
CCS Concepts

• **Security and privacy → Web application security**; • **General and reference → Measurement**.

Keywords

Web privacy measurement, Web crawling

ACM Reference Format:
Karthik Ramakrishnan, Qinge Xie, Uma Anand, and Frank Li. 2026. Crawling in the Deep: Evaluating Web Crawling Configurations for Web Privacy Measurements. In *Proceedings of the 2026 ACM Internet Measurement Conference (IMC '26), October 12–16, 2026, Karlsruhe, Germany*. ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3777912.3839794



IMC '26, Karlsruhe, Germany

ACM ISBN 979-8-4007-2327-8/2026/10
https://doi.org/10.1145/3777912.3839794

1 Introduction

Web measurements have been used over the years to study the privacy-related behaviors of websites at scale, such as cookies [14, 20, 31] and fingerprinting [32, 35]. These measurement studies have largely relied on web crawling techniques to conduct their analysis [11–13, 26], but crawling configurations vary widely across studies in terms of the parameters considered. Researchers have made different choices about how to navigate a website's internal pages, how many internal pages to visit per website, whether to simulate user interaction, whether to maintain browser state between visits, and how long to wait for a page to load. These methodological differences introduce challenges in interpreting, comparing, and reproducing the measurement outcomes across studies. More importantly, the implications of their crawling configuration choices have not been systematically evaluated, leading to questions of how representative and comprehensive the privacy measurements are.

The impact of using different crawling configurations has not gone unnoticed in other contexts. Prior studies [3, 4, 11, 12, 45, 50] have systematically evaluated how various crawling setups affect measurement results, but they mainly focused on non-privacy-related metrics collected during the crawl, such as code coverage [45], the structure of HTTP request chains [12], and the number of web objects and resources loaded [4]. Where prior work does consider privacy metrics, it varies the measurement environment rather than the crawler itself, comparing browsers and devices [9] or vantage points and consent states [37, 43]. Currently, there is a lack of systematic evaluation regarding how the design of the crawler itself, such as how it navigates a site and how long it observes each page, shapes the privacy behaviors observed. This gap is particularly salient as privacy behavior is fundamentally tied to the pages and content visited on sites, in contrast to many other website properties that are uniform across pages (e.g., TLS setups, web server configurations). Furthermore, web privacy measurements

are rather unique in their evaluation of often hidden and adversarially obscured behavior, rather than more typically observable properties such as performance or security policies.

To fill this gap, our work takes a structured approach by performing controlled web crawling experiments across the top 10K CrUX websites using 12 distinct crawling configurations. We vary configuration parameters including *search strategy* (breadth-first, depth-first, hybrid), *browser state* preservation (stateless, stateful), and simulated *user interaction* as part of the crawling configurations. For each configuration, we also evaluate the impact of the *search depth* of the search strategy and the *crawl timeout per page*, which determines how long the crawler stays on each page. For crawls under these various configurations, we monitor and analyze three key privacy-related metrics – HTTP-requested domains, Javascript Web API accesses, and cookies – to understand how the design of the crawler configuration affects privacy measurement outcomes.

From our analysis, we identify a crawl search strategy that finds broader sets of internal pages and drives higher observed privacy metrics. We also identify that statefulness and interaction do impact crawl outcomes, although more so for cookies than the other metrics. We additionally propose a dynamic strategy for determining search depth on a domain, which compared to using a static search depth threshold, provides high privacy behavior coverage while substantially reducing the number of internal pages crawled across domains. Finally, we identify the effectiveness of shorter page timeouts, offering opportunities for reducing runtimes for deep crawls.

Our findings lead to actionable recommendations for researchers conducting web privacy studies, highlighting trade-offs and best practices in crawler design. In summary, our work provides the following contributions:

- We conduct a large-scale experiment that systematically compares 12 distinct crawling configurations for web privacy measurements. These configuration combinations consist of multiple parameters including search strategy, browser state, user interaction, search depth, and crawl timeout per page.
- We evaluate the crawling configurations based on three privacy metrics: HTTP-requested domains (including tracking domains), JavaScript web API invocations (including fingerprinting APIs), and cookies (including tracking cookies). This allows us to understand how different configurations affect various potential privacy metrics collected.
- Based on our findings, we provide clear recommendations on which crawler configurations offer the best trade-offs between practicality, reproducibility, and efficiency based on different limitation scenarios. Our work helps researchers make informed decisions when designing future privacy measurements based on their requirements and goals.

2 Related Work

Here, we discuss related work on the use of crawlers in web privacy measurement and crawler evaluation.

Web Privacy Measurement. Web crawlers are vital in web privacy measurement studies to gather privacy-related metrics. Prior research has used them to investigate cookies [14, 20, 31], fingerprinting [32, 35], as well as other web tracking topics, such as link decoration [30], anonymized tracking data [15], and differential tracking across news websites [49].

To collect specific web metrics, these studies typically select a set of domains (often from top lists) and use web crawlers (commonly OpenWPM [19]) to visit each domain's landing page as well as several randomly chosen subpages, conducting either stateless or stateful crawls, with or without interaction. For example, to measure link decoration abuse, Munir et al. [30] used a stateless crawling strategy that randomly scrolls and moves the cursor while visting the landing page and selects 20 random internal pages from the landing page for further crawls. As another example, Demir et al. [14] performed a stateful crawl for each target site, investigating cookie banners while visiting the landing page and 15 randomly selected subpages from the landing page. However, variations in these crawler configurations can potential lead to differences in the observed metrics, raising questions about crawling strategies.

Crawler Evaluation Studies. The impact of web crawling configurations has not gone unnoticed, with prior studies evaluating how various setups affect measurement results. As discussed, prior studies using web crawlers often visit both landing pages and internal subpages to capture diverse site behaviors. Thus, a portion of prior work [4, 16, 46, 49] mainly focused on evaluating the differences between website landing pages and their internal pages. For example, Aqeel et al. [4] compared the landing pages of 1K sites with their internal pages and showed that internal pages differ substantially from landing pages in both content and performance. They also showed that internal pages can vary significantly from one another. In this study, we consider both landing and internal pages as well, and evaluate different internal page crawling strategies.

Other prior work [3, 50] has conducted evaluations of different crawling configurations across various non-privacy-related contexts. Meanwhile, Stafeev et al. [45] systematized the crawling methodologies of 403 papers that use web crawlers in security, privacy, and empirical web measurements. They identified commonly used navigation algorithms include both randomized and non-randomized BFS and DFS, as well as rule-based and search engine methods for selecting internal pages. They then evaluated these approaches on 2K domains randomly selected from the Google CrUX top 10K domains, considering three metrics as performance indicators: code coverage, JS source coverage, and link coverage. They found that randomized search algorithms generally performed better across all three metrics, and that increasing the search budget yields diminishing returns (which align with our results). In this work, we look at BFS, DFS, and breadth-limited BFS (all "randomized" under the prior work's definition). We exclude rule-based as they are custom strategies that are designed for specific measurement objective (e.g., finding OAuth 2.0 endpoints on the webpage [8]). We do not employ search engine methods for collecting internal pages as exploring a website via various crawling algorithms is the core focus of our study.

Demir et al. [11] focused on evaluating the reproducibility and replicability of web measurement studies conducted with different crawling setups, and showed that slight variations in those setups can directly impact the overall results. They surveyed 117 research papers and performed experiments to cross-compare 24 measurement setups on 4.5M pages, including variations across four

browsers, three regions, and with and without user interactions. Another work by Demir et al. [12] focused on investigating the causes of differences in the outcomes of web measurement experiments due to the crawling setups. They evaluated five measurement setups on 25K sites from the Tranco top list, differences in browser version, user interaction, and whether they operated in headful or headless mode of the browsers. By comparing the tracking request trees as an indicator, they found that the trees differed considerably across setups.

A separate line of work evaluated privacy metrics while varying the measurement environment. Cassel et al. [9] compared tracking and fingerprinting across real desktop and mobile browsers; Singh et al. [43] mapped tracker flows varying across 23 countries; and Prasad et al. [37] investigated the impact of the browser, vantage point, and consent state. Hantke et al. [23] instead targeted the measurement tooling, arguing that customized instrumentation negatively impacts accuracy and reproducibility.

Unlike these prior evaluations, which target general indicators such as code or link coverage and tracking-tree similarity, or vary the measurement environment rather than the crawler, our work evaluates crawler design parameters directly against privacy-specific metrics, proposes a dynamic per-site termination strategy that adapts to when each metric saturates, and provides recommendations for future web privacy measurements. Our findings on search strategy and user interaction align with prior observations [11, 12, 45] on a new family of metrics, while browser state, per-page timeout, and dynamic termination have not previously been evaluated in this way.

3 Method

Our work investigates web crawling configurations when used for web privacy measurements, by experimenting with these various configurations and monitoring their impact on web privacy metrics as indicators. In this section, we describe the crawling configurations evaluated and the design of our experiments. We consider various crawling parameters, including the page navigation strategy, the state of the browser between page visits, interactivity with pages, the amount of time spent on each page (crawl timeout), and the crawl depth. We then evaluate different crawling configurations constructed from various combinations of these crawl parameters.

We evaluate a total of 12 crawling configurations, as shown in Table 1, derived from the combinations of three dimensions: *Search Strategy, Browser State* and *User Interaction*. For each crawling configuration, we use constant values for the *Crawl Timeout per Page* and *Search Depth*. These constant values serve as the upper bounds in our evaluation of the two parameters, as our measurement process continually tracks the collected privacy metrics as these two parameters are increased, enabling us to analyze the impact of varying them from small increments up to the configured constant.

3.1 Crawling Parameters

We begin by describing the crawling parameters we evaluate and the values considered for each. Throughout this work, a crawling configuration refers to these crawler design parameters, and not to the browser, the execution environment, or the region from which a crawl is run. To ensure comparability, we hold these other aspects constant across all crawls, using the same operating system, browser configuration, docker containers for deployment, and network vantage point. Doing so isolates the effect of how a crawler explores a site, and also ensures that websites do not treat our crawlers differently based on device or platform-specific optimizations. We utilize a comprehensive set of parameters considered by prior works [11, 12, 45] in our evaluation of crawling configurations.

No.	Crawling Config Name	Search Strategy	Browser State	User Interaction
1	BFS-noS-noI	BFS	Stateless	No
2	BFS-noS-I	BFS	Stateless	Yes
3	BFS-S-noI	BFS	Stateful	No
4	BFS-S-I	BFS	Stateful	Yes
5	DFS-noS-noI	DFS	Stateless	No
6	DFS-noS-I	DFS	Stateless	Yes
7	DFS-S-noI	DFS	Stateful	No
8	DFS-S-I	DFS	Stateful	Yes
9	LBFS-noS-noI	Breadth-limited BFS	Stateless	No
10	LBFS-noS-I	Breadth-limited BFS	Stateless	Yes
11	LBFS-S-noI	Breadth-limited BFS	Stateful	No
12	LBFS-S-I	Breadth-limited BFS	Stateful	Yes

Table 1: Overview of the 12 crawling configurations evaluated in the study.

3.1.1 Search Strategy. The web page search strategy (or navigation algorithm) is the algorithm researchers use to navigate a website's internal pages, starting from its landing page. Prior work [45] systematized existing web measurement studies and identified that the choice of search strategy could significantly impact measurement outcomes, such as code coverage. Thus, we consider search strategy as one of the crawling parameters to systematically analyze its effect on the collected privacy metrics. In this work, we consider three fundamental search strategies systematized in prior work [45]:

- **Breadth-First Search (BFS)**: BFS starts at a site's landing page and systematically explores its internal pages level by level. It first visits internal pages directly linked to on the landing page, and then visits further internal pages that are directly linked to pages within the first level, before progressing to subsequent levels.
- **Depth-First Search (DFS)**: Starting at the landing page, DFS selects one internal page link and visits that link, and then recurses on that internal page (following one link on that page). This strategy supports backtracking, where if there are no new internal links on the current page, DFS returns to the previously visited page (and continues backwards until a new internal link can be found).
- **Breadth-Limited BFS**: This strategy is a hybrid approach between BFS and DFS, where one performs BFS but limits its branching factor to promote deeper searches. Specifically, breadth-limited BFS restricts how many internal links you explore per page. This strategy also supports backtracking, where if no new internal

links are available, the search can continue with previously excluded links on previously visited pages.

We observe that 50 pages is the highest cutoff for collecting internal pages in prior work [45] . Thus, we explore each search strategy on up to 50 active pages. Note, 50 pages is the upper bound for our evaluation and we will evaluate the effect of shallower search depths in Section 4.3. For the breadth-limited BFS, we evaluate a branching factor of 3, providing reasonable search depth given the 50 page limit (we leave it to future work to explore further branching factors). Meanwhile, since many internal links on websites may be inactive (e.g., HTTP 404), a crawl/search may generate numerous requests before it finds the 50 active pages. Thus, we cap the total number of pages requested at 100 per site, both to limit the network traffic sent to a site (for ethical considerations) and to bound the total computational effort expended in a realistic crawl.

3.1.2 Browser State. While performing web crawls, the browser's state can be preserved between different page visits of the same domain (Stateful), or cleared after every page visit (Stateless). Demir et al. [11] surveyed 117 web measurement papers and found that 41% of them do not discuss the browser state in their crawling configurations, and most of the remaining studies relied on stateless crawls [4, 7]. We explore browser state in this study as it can result in the pages of a website behaving differently, such as the saved cookies, and it is also closely tied to advertising, fingerprinting, and user tracking behaviors, thus may impact privacy-relevant behaviors [33].

3.1.3 User Interaction. Prior work [11, 12] has shown that mimicked user interaction could significantly impact the embedded objects and third-party content. Thus, we explore the use of simulated user interactions on pages. We use a modified version of the simulation method adopted by prior work [11, 12]. Their method entails initiating Page Down, Tab, and End keystrokes at random intervals after the page has completely loaded before simulating these interactions.

In our experiments that enable page interaction, we start simulating user interactions immediately upon initial page load. We send the Tab keystrokes with short periods of random delays as used by prior work [12, 26]. In addition, we scroll to the page bottom every five seconds, which is the time interval we use later during the analyses, calculating the *scrollHeight* of the webpage's body to determine the amount of scrolling. Compared with prior methods that relied upon Page Down and Page Up keystrokes, we apply window scrolling to add diversity to the types of user interactions simulated. Furthermore, scrolling to the bottom of the page can cause further dynamic rendering of the page, such as on popular sites like Facebook and LinkedIn. This additional web content can generate further behavior that impacts our privacy metrics. We note that we avoid interactions such as page clicking or element interaction, as these can trigger navigations to other pages. Our crawler navigates by extracting the internal links on a page and requesting their URLs directly, so following links does not require clicking page elements. This decision also means we did not interact with any cookie consent banners, but this state is equivalent across all configurations and reflects the default behavior of the website pages.

3.1.4 Search Depth. Search depth is the number of internal pages crawled per domain, where our cutoff is set at 50 internal pages (based on prior work, as discussed in Section 3.1.1). While searching deeper may uncover pages with new privacy-relevant behavior, it also entails additional computational resources. We explore each search strategy up to 50 pages, and aim to understand the cost/benefit tradeoff for different search depths.

3.1.5 Crawl Timeout per Page. Finally, we consider the crawl timeout per page, which is the duration a crawler remains on a page before navigating to other pages. Staying on a page longer may capture more of its dynamic behaviors, such as the execution of JavaScript scripts. Most prior work [11, 12, 20, 26, 46, 47] used a crawl timeout of up to 30 seconds in their measurement setups. Thus, we adopt 30 seconds as the constant value for the timeout parameter, allowing us to evaluate its impact across a range of intervals from small increments up to 30 seconds. As with search depth, 30 seconds is an upper bound for our evaluation rather than a fixed value, and we evaluate the effect of shorter timeouts in Section 4.4.

3.2 Crawling Metrics

To evaluate crawling configurations, we examine their impact on a set of privacy metrics derived from prior web privacy measurement studies. The selected metrics, HTTP-requested domains, JavaScript APIs, and cookies, are extensively studied [12, 14] privacy indicators covering various privacy analyses such as third-party interactions, fingerprinting, and user tracking. We also analyse a privacy-relevant subset of these metrics, since the top-level metric is related but not directly about privacy.

- **HTTP Requests:** We capture all HTTP requests for the visited webpages, including the resources fetched by the webpage (e.g., images, HTML files, and JavaScript). We also classify whether these requests are tracking-related by using blocklists compiled from prior works [11, 12, 19, 25, 35, 47]. The list of blocklists used are Easylist [44], EasyPrivacy [17], and Fanboy's Social/Annoyances/Notifications Blocking List [18]. Instead of considering full URLs, we focus on the unique second-level domains (SLDs) present in our dataset. This approach allows us to more clearly capture the diversity of third-party domains contacted by a webpage.
- **JavaScript Web APIs:** We capture all JavaScript web API invocations by scripts loaded on the webpage. Web APIs can be used for different functionalities, including potentially privacy-invasive actions like obtaining geolocation of the device, camera access, or microphone access. Prior works [1, 2, 10, 24, 42, 48] analyzed browser fingerprinting techniques and identified commonly used web APIs. These APIs are detected by OpenWPM's fingerprinting detector [34], which we use to classify a privacy-relevant subset of web APIs.
- **Cookies:** We capture all cookies being set, modified, or deleted by scripts which are executed on the webpages being crawled as well as by HTTP response headers. We consider unique cookie name-domain pairs as they are consistent and comparable across different sites as compared to cookie values, which are frequently overwritten and set to unique randomized values. Our analysis covers both first-party and third-party cookies. Prior works [5, 21, 31, 33, 39, 41] have shown cookies being used for cross-site

tracking purposes, as well as browser fingerprinting being used to set persistent cookies with specific identifiers [22, 28].
Thus, we further classify cookies being used for tracking purposes using the machine learning classifier developed by Bollinger et al. [5], which categorizes cookies by their purposes into four categories: Necessary, Functional, Analytics, and Advertising/Tracking. We take the Analytics and Advertising/Tracking categories together as our explicitly privacy-relevant subset. Our aggregate cookie-based results exhibit the same high-level trends regardless of this categorization, indicating that our findings are not sensitive to the specific cookie grouping used.

3.3 Measurement Framework

Here, we describe our measurement framework, including our experimental sites, crawling design and setups.

3.3.1 Websites Dataset. To perform meaningful evaluation of the crawling configurations, our measurement study requires a set of websites to crawl, which includes selecting from a top list. The Chrome User Experience Report (CrUX) top list [40] is widely used and considered as a more accurate representation of website usage compared to other DNS-based top lists [40], or the aggregated Tranco list [36]. Thus, we crawled the top 10K websites from the CrUX dataset generated in October 2024, which includes sites in the top 1K, 5K, and 10K rank buckets.

3.3.2 Crawling Design. Our goal is to evaluate how different crawling strategies influence the internal pages collected and, in turn, the privacy metrics discussed in Section 3.1. The search strategy is the only factor that affects which pages are gathered; other configuration parameters do not change the set of pages. Consequently, configurations that share the same search strategy are evaluated on the same set of pages, allowing for a consistent comparison of their effects. As discussed in Section 3.1, we consider 12 crawler configurations in our analysis to study the impact of these strategies on the collected data. All of our measurements are performed for these configurations, which are then evaluated based on the privacy metrics collected. We split our measurement into two phases: Pages collection phase and Parameters evaluation phase.

- **Pages Collection Phase**: In this phase, we crawl a website using a specific search strategy and record the ordered set of pages that are visited during that crawl (up to our crawl depth cap). This initial crawl is performed using a stateless configuration without user interaction (using the max crawl timeout per page), for all of the search strategies. The resulting set of pages serve as the basis for subsequent evaluations of the other crawling configurations under the same search strategy (in the next measurement phase). Re-using these same pages crawled with differing parameters (e.g., browser state and user interaction) affords direct comparisons (whereas if each crawl visited different pages, the variations in privacy metrics could be due to those pages rather than the crawling parameters).
- **Parameters Evaluation Phase**: In this phase, we evaluate and compare all crawl parameters, by crawling the same set of pages identified in the prior phase (for a given search strategy) under different crawl configurations (e.g., browser state and user interaction). During each crawl, we record all relevant events as described in Section 3.2 (i.e., HTTP requests, Web API calls, and cookie operations) and their timestamps, allowing us to track the different privacy metrics across pages and time on pages. We again crawl to a max search depth and use a max crawl timeout per page, as we can evaluate lower depths and timeouts by filtering out events generated after a target depth or timeout.

3.3.3 Crawling Setup. We use OpenWPM (v0.30.0) [19], a popular crawling framework for privacy measurements coupled with the Firefox (v130.0) browser. We run the browsers in headful mode with an Xvfb display to best represent real web browsing traffic and to avoid being classified as bots.

We retain OpenWPM's default browser configuration, and do not alter Firefox's default privacy settings ourselves. The one privacy-relevant preference that OpenWPM sets is the cookie policy, which accepts all cookies, including third-party ones. It uses the default Firefox's tracking protection preferences, does not send the Do Not Track header, and does not install content blocking or anti-fingerprinting extensions. We instrument HTTP requests and responses, cookie changes, navigations, JavaScript Web API calls, and DNS resolutions. We adopt this default configuration so that our comparison reflects the crawling parameters rather than the coverage of a particular blocklist, and it is held constant across all 12 crawling configurations so it does not bias our comparisons.

OpenWPM has been widely adopted in web privacy measurement research [7, 11, 12, 14, 32] and provides a comprehensive set of privacy metrics by default, instrumenting all three of our metric families within a single framework. Using an established setup also allows our results to be checked against prior work and easily replicated. OpenWPM records the timestamps for all of the events related to the privacy metrics discussed in Section 3.2, allowing us to analyze if and when certain metrics change.

Our crawlers were distributed across 5 different servers hosted in an academic institution's network in the US. For crawls under stateless configurations, we scheduled page visits per domain such that each domain was visited at most once per minute (multiplexing our crawlers across domains, clearing browser state after every page visit). For stateful crawls, the same crawler (and the browser instance) consecutively crawled the pages for a domain, while still maintaining the same page visit rate, without clearing browser state until all domain pages were visited. The rate limiting of page visits was both for ethical considerations (to limit the network load induced on sites) and to avoid being classified as bots due to overly frequent requests.

As part of the Pages Collection Phase, we were successfully able to crawl 9,673 of the top 10K sites. We manually investigated crawl failures and identified they were due to the site/landing page being offline, or being blocked by bot management systems (e.g., Cloudflare interstitials, CAPTCHA gates, or 403/503 responses), rather than errors with our measurement execution. These failure modes affect all configurations equally and thus do not bias our cross-configuration comparison.

3.3.4 Measurement Stability Evaluation. To evaluate the stability of our results over time, we conducted a separate round of smaller-scale measurements in March 2025 (5 months after our original experiment). We assessed 1K randomly selected sites from the same top 10K sites used in the original experiment, and collected a fresh

set of internal pages for evaluation. The separate round allows us to evaluate the impact of both time and page selection on our findings.

3.4 Limitations

Before analyzing results, we discuss our study's limitations. First, like prior web measurements [11, 12, 29, 31], we use OpenWPM with the browser configuration reported in Section 3.3.3, which includes a comprehensive set of instrumented web APIs. We do not instrument additional web APIs in this work, such as the Battery Status API, which should minimally impact our results but could be incorporated by future work.

Although we mimic user actions on a page, we do not directly interact with page elements (e.g., clicking a button) to avoid unintended consequences, such as potential being redirected to other pages. Our measurements do not include the investigation of post-login/authenticated behaviors. Prior work [27, 38] typically relied on manually created accounts to evaluate a small number of sites, and this approach is challenging to apply in our large-scale measurements. Thus, our privacy metrics may undercount the true privacy behaviors on a site. However, we note that this limitation applies equally to all crawl configurations evaluated, and does not bias our comparisons. Furthermore, web measurements in practice are likely restricted to such design decisions for similar reasons.

We do not interact with cookie consent banners, so we observe each site in its pre-consent state, reflecting its default privacy behavior when a user does not act on a consent notice. All 12 configurations observe this same state and thus remain comparable, though behaviors may differ once a user accepts or rejects consent, which we leave to future work.

Our crawls may also be subject to bot detection. Beyond the crawl failures reported in Section 3.3.3, sites that fingerprint OpenWPM may serve degraded content partway through a crawl (we do not enable OpenWPM's bot mitigation), which is difficult to distinguish from genuinely saturated privacy metrics. This affects all configurations equally and so should not bias our comparisons, though it may reduce the absolute amount of behaviors observed.

We consider browser state only within a site, preserved or cleared across pages of the same domain. Preserving state across domains raises complexities with which domains to preserve state across and the order of accessing domains, making it a separate experimental axis that we leave to future work.

Our measurements were performed from five co-located servers on one academic ISP in the state of Georgia, US. Prior work [11] has noted that changing web crawling measurement locations could also impact crawling results, including via region-specific filter lists [6]. Thus, our results may differ for crawls run from other US states or countries, which we leave to future work.

4 Results

Through our data collection method, we produce a dataset that tracks our privacy metrics (Section 3.2) throughout multiple crawls of a website under different crawling configurations. This dataset allows us to directly compare how the crawling parameters impact the privacy metrics observed, at scale across websites. Here, we analyze our experimental data to identify the impact of these various crawling parameters on web privacy measurements.

4.1 Crawling Coverage

First, we evaluate how the search/navigation strategies differ in the number of responsive pages crawled per site. Recall that in our measurements, we crawl up to 50 responsive pages per site (using backtracking when relevant during navigation), but also capped the number of attempted page requests at 100 (regardless of whether the page request succeeded). In Figure 1, for each search strategy, we plot the distribution of the number of responsive pages crawled per site. We note that for nearly 10% of the sites, there was only 1 responsive page found, which was the landing page. We manually investigated a random sample of 100 such sites, and confirmed that they either redirected to content delivery networks or advertisement networks (as also observed in prior work [12]), or indeed contained no first-party links on the landing page.

We observe that overall, BFS visited more responsive pages than the other two strategies, with DFS visiting the fewest. With BFS, 50 responsive pages were found on approximately 79% of sites, while for breadth-limited BFS we were able to fully crawl for around 67% of sites. In comparison, DFS did not reach 50 live pages on nearly half of the sites (meaning for such sites, we exceeded 100 pages attempted yet still did not identify 50 live pages). While these caps on crawling depths are specific to our experiment, they demonstrate that DFS and to a lesser extent, breadth-limited BFS, are less efficient at finding live pages to evaluate compared to BFS. Intuitively, this indicates that internal links on pages deep within a site (i.e., those explored more by DFS) are less likely to still be available compared to those more closely linked to the landing page (i.e., those explored more by BFS).

While BFS may be more effective at finding live pages, that does not necessarily mean that the pages found provide more comprehensive visibility into privacy behaviors. Thus, we next explore how the different strategies compare across our privacy metrics.

> **Takeaway:** BFS is more efficient at finding live pages on a site as compared to DFS and breadth-limited BFS search strategies.

4.2 Crawling Strategies

Here, we analyze how the search strategy, browser state, and interaction crawling parameters impact the privacy metrics collected (we investigate the impact of search depth and crawl timeout per

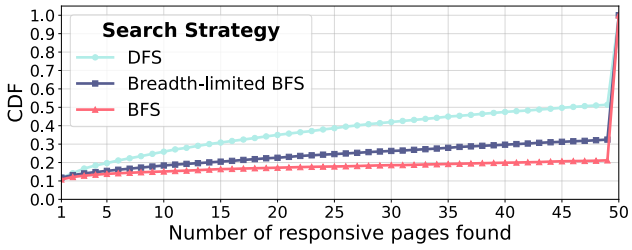


Figure 1: CDF of the number of responsive pages crawled per site, using different search strategies. Crawls were capped at 50 responsive pages, and up to 100 attempted page requests.

page in more detail later on). For this analysis, we use a full 30 seconds crawl timeout per page, to maximize what could be observed in our experiment for each crawl configuration.

For each privacy metric, we examine both the total quantity and the per-site distribution of the metric collected across different crawler configurations, evaluated at different depths. Evaluating these together allows us to understand not only how much privacy-relevant data is captured in total, but how the data collected is distributed across sites. We also center our analysis on the privacy-specific metrics (i.e., HTTP-requested tracking domains, fingerprinting web APIs, and ad/tracking cookies), but we briefly discuss the impact on the more general metrics (i.e., all HTTP-requested domains, web APIs, and cookies).

We use non-parametric statistical testing to evaluate differences between crawling configurations in the metrics observed across sites (as our distributions are not normal). Specifically, we use a pairwise Wilcoxon signed-rank test to compare two configurations, which is a paired test as our measurements are on the same population of sites. Our default threshold for significance is $\alpha = 0.05$. Since we perform multiple comparisons between various pairs, we apply Holm-Bonferroni multi-test correction, which adjusts the significance threshold as more comparisons are performed to maintain a constant error rate across all tests.

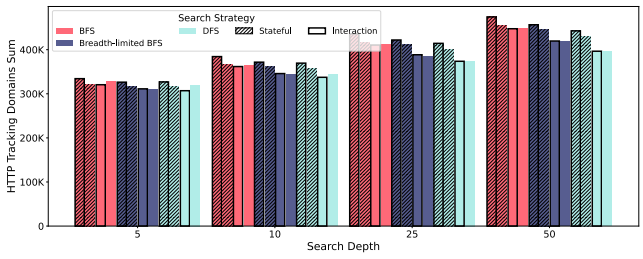


Figure 2: Sum of the number of unique HTTP-requested tracking domains observed per site, aggregated across all sites, when searching to varying depths.

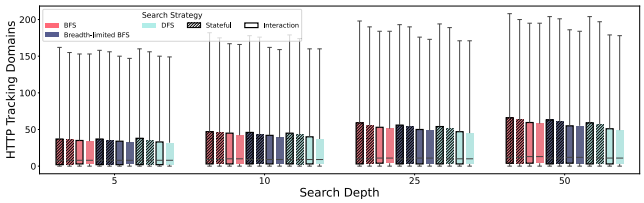


Figure 3: Number of unique HTTP-requested tracking domains observed per site, when searching to varying depths. The top whisker represents the 95th percentiles.

HTTP Requests. We start with analysis of the number of unique second-level domains (SLDs) observed in HTTP requests when crawling each site, centering our analysis on the tracking domains (as our privacy-specific metric). Figure 2 shows the total number of unique tracking domains observed per site, summed across all sites (uniqueness is per site, not across all sites), with varying crawl search depths and configurations. In essence, this reflects the aggregate metric observed across an entire crawl measurement. We observe that BFS configurations consistently yielded higher totals than both DFS and breadth-limited BFS, ranging from 1–14% more total tracking domains than equivalent crawls with alternative search strategies, across all depths. For browser state and interaction, we generally see that stateful crawls and interactive crawls drove higher metrics, and their combination resulted in the highest sums regardless of search strategy and depth. Across depths and search strategies, the stateful with interaction crawl was between 0.8–11.6% higher than other crawls under the same search strategy. Overall, we find BFS stateful with interaction produced the highest aggregate metrics regardless of depth, from 4.3-6.3% higher than the second best configuration.

We next evaluate the distribution of unique tracking domain counts observed per site, as shown in Figure 3. This analysis provides visibility into whether aggregate differences are due to a small number of outlier sites or represent broader distributional differences across sites. First, we observe that at all depths, the median number of tracking domains observed per site is similar across all configurations. However, there are more noticeable variations at the upper quartiles of the distributions, with BFS, stateful, and interactive configurations skewing towards more tracking domains per site, particularly at deeper crawls. For example, at depth 50, the 75th percentile site during a crawl using BFS stateful with interaction showed 2.6–34.7% more tracking domains than those of other crawling configurations, aligning with the aggregate analysis.

Evaluating whether these distribution differences between crawling configurations are statistically significant based on pairwise Wilcoxon tests (with correction), we observe statistically significant differences between nearly all configuration pairs across all the search depths, including between BFS stateful with interaction and all other configurations across all search depths. These results signal that crawling configurations do notably impact metrics based on HTTP-requested tracking domains.

Finally, we also evaluate all unique HTTP-requested domains per site (our generic metric), rather than just tracking domains (our privacy-specific metric). While this metric is not directly privacy-related, it is a superset of tracking domains so may correlate with other privacy metrics related to HTTP-requested domains. Appendix Figures 13 and 14 present the aggregate and per-site distributions of all unique HTTP-requested domains, respectively. We again observe the same patterns, that BFS, stateful, and interaction each drove higher metrics, with the combination of all three resulting in the most performant configuration. For aggregated metrics across sites, BFS stateful with interaction observed between 4.8 – 9.0% more unique domains than other configurations, averaged across all depths, and this difference is statistically significantly different than other configurations across all depths.

Web APIs. We next investigate Web APIs accessed during crawls, focusing first on fingerprinting APIs as a privacy-specific metric. Figure 4 shows the aggregate, and Figure 5 the per-site distribution. The same pattern holds as with HTTP-requested domains, with BFS, stateful, and interactive crawls driving higher counts, but margins are smaller because the set of Web APIs is finite: BFS stateful with interaction leads aggregate counts by 0.5–1.2% over the next-best

configuration, and per-site distributions converge across configurations even at upper quartiles. Despite smaller margins, pairwise Wilcoxon tests (with correction) show BFS stateful with interaction is statistically significantly different from all other configurations at all depths, except BFS stateless with interaction at depth 50.

For all Web APIs (a superset including non-fingerprinting; Appendix Figures 15 and 16), the same trends hold: BFS stateful with interaction yields 0.9–8.0% more in aggregate than other configurations, with the same statistical testing exception (BFS stateless with interaction).

Cookies. Finally, we investigate Cookies set either by scripts or via HTTP response headers while crawling each site. As in previous metrics, we start by characterizing cookies that are explicitly flagged for advertising or tracking purposes as a privacy-specific metric. Figure 6 shows the total number of unique tracking cookies

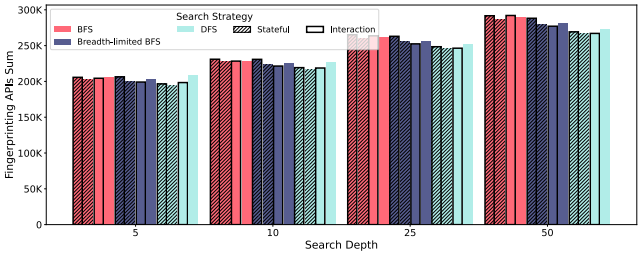


Figure 4: Sum of the number of unique Fingerprinting APIs observed per site, aggregated across all sites, when searching to varying depths.

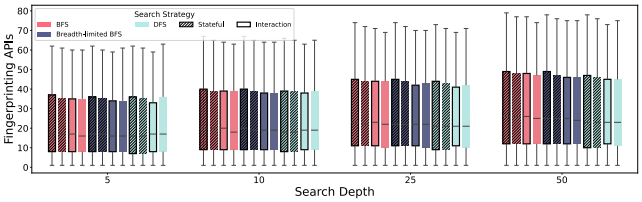


Figure 5: Number of unique Fingerprinting APIs observed per site, when searching to varying depths. The top whisker represents the 95th percentiles.

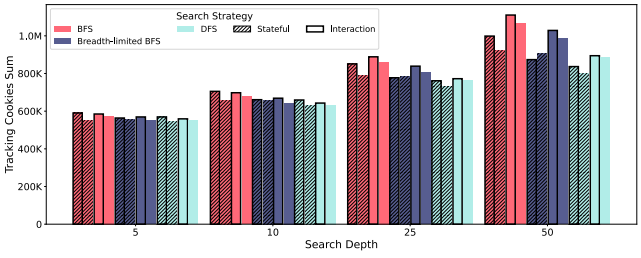


Figure 6: Sum of the number of unique tracking cookies observed per site, aggregated across all sites, when searching to varying depths.

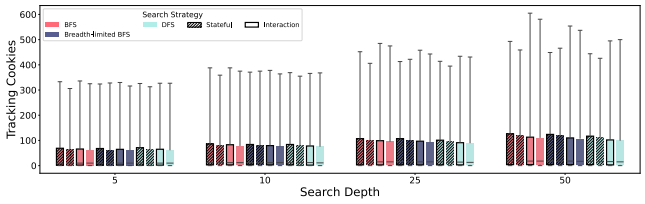


Figure 7: Number of unique tracking cookies observed per site, when searching to varying depths. The top whisker represents the 95th percentiles.

observed per site, summed across all sites (uniqueness is per site, not across sites), with varying crawl search depths and configurations.

We observe that BFS configurations yield consistently higher totals for tracking cookies than DFS or breadth-limited BFS, collecting 1–26% more tracking cookies compared to equivalent crawls under different search strategies, across depths. While interaction likewise increased the aggregate total as with the prior metrics, browser state offered different behavior here. In particular, stateless crawls resulted in the highest aggregate metrics. For example, at depth 50, BFS stateless with interaction exhibited the highest totals, 9.5% higher than the second best strategy (which was BFS stateless without interaction).

Figure 7 shows the per-site distribution of tracking cookies. Here, our observations do not align with the aggregate analysis. In particular, the median number of tracking cookies per site is similar across all configurations, and we see higher 75th percentiles for BFS, stateful, and interactive crawls. Here, unlike with the aggregate perspective where stateless crawls performed best, the distributions for stateful crawls skew higher than for stateless ones. Investigating these conflicting observations by manually analyzing sites, we identified that a small number of sites exhibited abnormally high numbers of cookies when stateless, as they set a large number of new, uniquely-named cookies per page during a stateless crawl (even though these cookies likely serve the same functions across pages). These sites are outliers in the per-site distributions, so do not notably shift the overall distributions, but contribute substantially to the aggregate totals. Thus, if raw numbers of unique cookies are most desired, BFS stateless with interaction crawls are most performant, but in practice, BFS stateful with interaction crawls drives broader observations of meaningful cookies across sites.

We again evaluate the statistical significance of differences in each crawl configuration's distributions, using Wilcoxon signed-rank tests using corrections. We find statistically significant differences with most pairs, including BFS stateful crawls with interaction compared to other configurations.

Finally, we consider all cookies collected (not only tracking ones), showing our aggregate and per-site analysis in Figures 17 and 18 in Appendix. We observe largely the same trends as with tracking cookies, where BFS stateless with interaction crawls result in the most total cookies, but the per-site distributions for BFS stateful with interaction crawls skew the highest.

Overall, these results suggest for cookies, BFS stateful with interaction crawls remains a top choice (although a stateless version can result in more total unique cookie counts).

Quality vs. Quantity of Discovered Pages. In Section 4.1, we found that BFS is more effective at crawling live, responsive pages, compared to breadth-limited BFS and DFS. Here, we saw that BFS-based crawls resulted in higher privacy-related metrics. Thus, we investigate whether BFS's advantage in privacy behavior collection is simply due to its advantage in finding responsive pages, to better understand the impact of this crawling parameter. (Note, even if this were the case, we would still recommend BFS-based crawls given their effectiveness at finding live pages.) To do so, we repeat our analysis on only the sites where each navigation strategy successfully discovered all 50 responsive pages. Thus, for these sites, all three navigation strategies were equally successful at finding responsive pages.

Again, we analyze the aggregate and per-site distribution of our privacy metrics across all of these sites, across varying search depth. For all metrics (both privacy-specific and general), even when constrained to fully-crawled sites, BFS-based configurations consistently exhibit a higher aggregate metric compared to breadth-limited BFS and DFS configurations, across all depths. This observation matches our original findings on the full set of sites. Thus, BFS's higher performance along these metrics is not only a consequence of its effectiveness at discovering more pages, but it must also generally crawl pages that are richer in privacy-invasive behavior.

Takeaway: BFS, stateful, and interactive crawls consistently provide the most unique observations across both our privacy-specific and general metrics. Thus, our results identify BFS stateful crawls with interaction as the most effective configuration overall.

4.3 Search Depth

Next we investigate how deeply to search during a crawl. Given our above finding that BFS stateful with interaction crawls are the most performant for the privacy metrics evaluated, for expository purposes, we focus here on analyzing this specific configuration. However, we observed that our high-level takeaways still apply to other configurations.

As we explore the relationship between search depth and privacy metric collection, we begin by investigating how additional exploration of internal pages uncovers more data. We first examine the impact of terminating the crawl at specific search depths, aiming to find the point where continued crawling yields diminishing returns. Beyond a static search depth threshold, we also explore a dynamic threshold where we terminate search only if crawling a certain number of consecutive pages does not yield changes in the privacy metric measured. To maintain an equivalent site population as we evaluate varying threshold values, here we focus on analyzing the sites for which 50 responsive pages were found.

4.3.1 Static crawl termination threshold. We start by analyzing the impact of crawling up to a static search depth on the amount of a privacy metric missed (compared to fully crawling up to 50 responsive pages). For a given depth threshold, we compute the metric *error rate* as the proportion of the total privacy metric count on a site (obtained from the full crawl) that is not captured by the cumulative count from crawling up to that depth threshold. For this

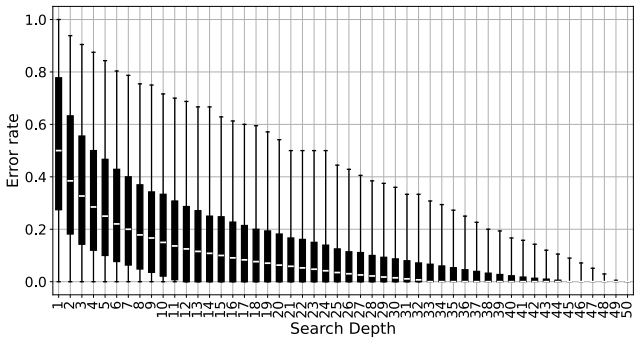


Figure 8: Boxplot of the per-site HTTP-requested tracking domain error rates, across sites and for different search depths, where the error rate is defined as the fraction of tracking domains missed when stopping the crawl at that depth, relative to the total domains found by a full crawl (up to depth 50). The top whisker represents the 95th percentiles.

section, we define saturation as the depth where a site reaches a 0% error rate (i.e., the privacy metric does not change crawling beyond that depth). Note that while our crawls terminated at a depth of 50, certainly one may observe new privacy behavior at further depths. Nonetheless, the analysis here will indicate how much value is typically yielded by additional pages crawled, providing insights into how deeply to crawl.

HTTP Requests. For HTTP-requested tracking domains (Figure 8), the median error rate reaches 0% at a search depth of 33; for general HTTP-requested domains, only at depth 39. A notable fraction of sites continue to exhibit non-zero error rates up to the maximum depth of 50, though the curve flattens as depth increases, indicating diminishing returns from deeper crawls.

Web APIs. For Fingerprinting APIs and general Web APIs, the median error rate reaches 0% at search depths of 23 and 24, respectively. This is earlier than HTTP-requested domains, reflecting the finite set of Web APIs typically used by sites; a minority of sites still surface new API usage up to depth 50. Web APIs can thus be efficiently captured by shallower crawls.

Cookies. Tracking cookies reach a 0% median error rate at depth 37, while overall cookies saturate later at depth 43; in both cases, a small fraction of sites continue to introduce new cookies up to depth 50, with diminishing returns from deeper crawls. This reflects how tracking cookies are typically deployed via shared third-party scripts (reused across pages), while other cookies may be tied to site-specific functionality at greater depths.

Takeaway: Different privacy metrics accumulate at varying rates across crawl depth, making it difficult to define a universal, static threshold for crawl search depth. Using a static, shallow threshold misses a significant portion of privacy behavior. For higher privacy behavior coverage, a high static threshold is needed (e.g., 40+ pages), but such a threshold is inefficient on many sites.

4.3.2 Adaptive Search Depth Threshold. The static crawl termination threshold explored above assumes all sites behave similarly,

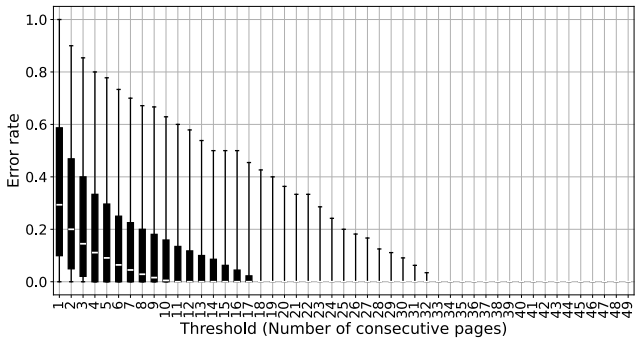


Figure 9: Boxplot of the per-site HTTP-requested tracking domain error rates, across sites and when terminating using various dynamic thresholds, where the error rate is defined as the fraction of tracking domains missed with dynamically terminating, relative to the total domains found by a full crawl (up to depth 50). The top whisker represents the 95th percentiles.

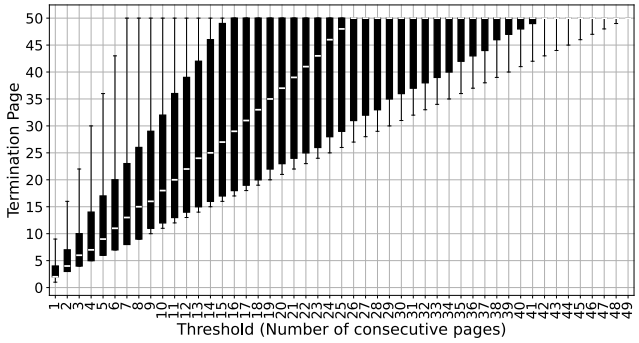


Figure 10: Boxplot of when crawls terminate on sites (the termination page) under the dynamic termination strategy, considering different thresholds for consecutive pages with unchanged HTTP-requested tracking domain count. The whiskers represent the 5th and 95th percentiles.

which is not the case in practice. Thus, our analysis showed that to capture most privacy observations using a fixed threshold, one would need to crawl deeply, often times more than necessary for many sites. Some sites expose most of their privacy-relevant behaviors within the first few internal pages, making additional crawling redundant. This observation motivates the need for an approach that is more adaptive to each specific site.

Instead of relying on a fixed threshold, we investigate a per-site dynamic threshold that adjusts based on how much the values of observed privacy metrics continue to change during the crawl. We explore an adaptive termination strategy where the crawl terminates if a metric (e.g., number of unique Fingerprinting APIs) remains unchanged after a threshold number of consecutive pages visited. This method aims to minimize crawling effort on sites where additional crawling provides diminishing returns (particularly early on in a crawl), while still allowing deeper exploration for sites where privacy-relevant behaviors continue to emerge throughout the crawl. However, this method may terminate crawls on some pages too early, where significant privacy behaviors are only observed deeper in the crawl. Thus, we investigate the effectiveness of the dynamic threshold approach.

To understand the trade-offs between crawl efficiency and completeness of privacy metric collection, we analyze two dimensions across various dynamic threshold values: 1) metric *error rates*, which we compute as the fractions of metrics missed by choosing a dynamic threshold value, relative to the total counts found by crawling to depth 50, and 2) the corresponding crawl depths.

HTTP Requests. For HTTP-requested tracking domains (Figures 9 and 10), a dynamic threshold of 11 yields a 0% median error rate[1] at a median termination depth of 20, crawling 178.7K pages in total. The corresponding static threshold of 33 requires 234.1K pages (31% more). The dynamic threshold is not uniformly better: the 95th percentile depth reaches 50, so a small minority of sites crawl deeper than under static.

For all HTTP-requested domains, a dynamic threshold of 11 yields the same 0% median error at a median depth of 22, crawling 202.9K pages, versus 276.6K under a static threshold of 39 (36.3% more).

Web APIs. For Fingerprinting APIs (Appendix Figures 19 and 20), a dynamic threshold of 12 yields a 0% median error at depth 18, crawling 100.7K pages, versus 110.7K under a static threshold of 23 (9.9% more).

General Web APIs show nearly identical behavior: a dynamic threshold of 13 yields 0% median error at depth 20, crawling 110.9K pages, versus 115.5K under a static threshold of 24 (4.2% more).

Cookies. For tracking and general cookies (tracking results in Appendix Figures 21 and 22), a dynamic threshold of 8 yields 0% median error, with median termination at depth 16 (tracking) and 22 (general). The corresponding static thresholds of 37 and 43 require 146.3K and 179.7K pages, respectively, versus 95.7K and 117.8K under dynamic (52.8% and 52.6% more).

Takeaway: Using a dynamic threshold for search termination on each site is significantly more efficient than a static threshold, and can result in high privacy behavior coverage while requiring far fewer pages visited across most sites.

For all metrics (both privacy-specific and general), a dynamic threshold of 13 pages resulted in a median 0% error rate (although a lower threshold of 8 worked best for cookies).

4.4 Crawl Timeout per Page

Next, we evaluate how per-page crawl timeouts affect the collection of privacy-related metrics. We consider crawl timeouts ranging

[1]Even if considering other error operating points, we generally observe that dynamic crawl termination is more efficient than using a static threshold. As an example, consider a more error-tolerant crawl seeking 75% of sites that exhibit an error rate lower than 10%. This is achieved with a dynamic threshold of 13, crawling 189.6K pages in total, versus a static threshold of 28, crawling 200.1K pages in total (5.3% more). Here, dynamic termination still remains more efficient than static thresholding, but the advantage is lower than other cases as the operating point is less error-sensitive. As seen in Figure 8, the error distribution with a static threshold flattens quickly as the static threshold increases, but does not reach zero error quickly. Thus, static thresholding performs relatively better when some error is tolerated.

from 5 to 30 seconds and measure when metrics are observed relative to page load time. For each timeout threshold, we compute an *error rate*, defined as the fraction of a metric missed when using that timeout compared to a full 30-seconds page timeout.

Here, we present results for search depth cutoffs of 5 and 50 pages, representing both shallow and deep crawls. As we will show, we observe similar findings at both depths, as well as at intermediate depths. Thus, the impact of the crawl timeout is consistent across search depths, and we elide discussion of intermediate depths.

HTTP Requests. We begin by analyzing HTTP-requested tracking domains. Figure 11 shows boxplots of per-site error rates for tracking domains across page timeouts between 5 and 30 seconds, for crawls with search depth cutoffs of 5 and 50 pages. Lower error rates indicate that most tracking domains observed with a 30-second timeout are already captured within shorter timeouts.

At a search depth cutoff of 5 pages, tracking domain error rates decrease rapidly as the timeout increases. As shown in Figure 11, the median error rate is 0% with 10 second timeouts, and 0% for 77.3% of domains with 15 second timeouts. As the timeout increases further, we see diminishing increases in error-free sites. We observe similarly at a search depth cutoff of 50 pages.

To confirm this rapid convergence, we examine when tracking domains are first requested during page load. For each unique (site, tracking domain) pair, we compute the earliest time after page load at which the tracking domain is requested on any crawled page of the website (as a tracking domain may be loaded by multiple crawled pages). Figure 12 plots the CDF of these earliest request times for search depths of 5 and 50 pages. Across both depth cutoffs, the distributions are similar: approximately 93–95% of (website, tracking domain) pairs are observed within 15 seconds after a page load, with nearly all observed by 20 seconds. Thus, tracking domains are indeed mostly requested within the first 15-20 seconds after page load (on at least one of the crawled pages), explaining why most website error rates approach 0% at those timeout thresholds.

General HTTP-requested domains follow the same pattern, with a 0% median error rate by 10 seconds and only 19.6% of sites

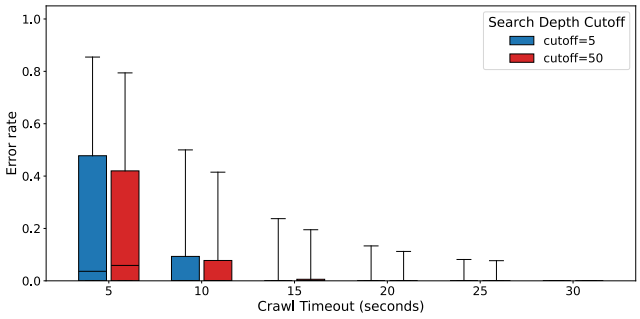


Figure 11: Boxplots of per-site error rates for HTTP-requested tracking domains, across sites, using varying page timeouts between 5–30 seconds. Error rate is the fraction of tracking domains missed relative to a 30-second page timeout. Crawls with search depths of 5 and 50 pages are shown. The top whisker represents the 95th percentiles.

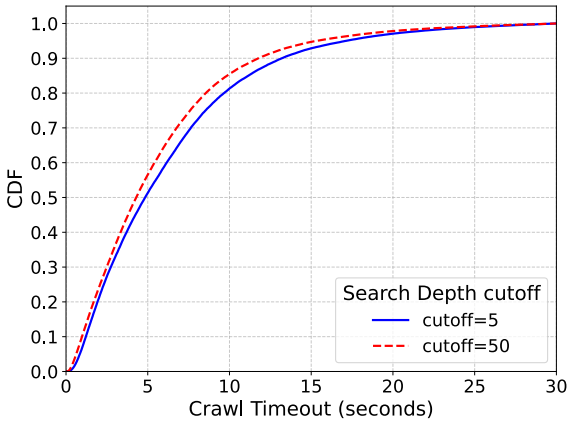


Figure 12: CDF of earliest observation times for HTTP-requested tracking domains on sites, computed using the earliest time after a page load that a tracking domain is requested on any crawled page of a site, across all (site, tracking domain) pairs.

with non-zero error rates at 15 seconds, indicating that most non-tracking HTTP requests also occur shortly after a page load.

Web APIs. We next examine Web API usage, focusing on fingerprinting APIs. The overall trends mirror those observed for HTTP-requested domains, but convergence is relatively faster. At a 5-second timeout, roughly half of sites already show 0% error across both search depths. By 15 seconds, nearly 83.8% of sites exhibit 0% error, and by 20 seconds this exceeds 87%, for both search depths.

General Web APIs exhibit nearly identical convergence behavior as fingerprinting APIs, with the fraction of sites showing 0% error at different timeouts within 1% of those for fingerprinting APIs.

Cookies. Finally, we analyze cookie-setting behavior, focusing on tracking cookies. At 10 second timeouts, the median site exhibits 0% error, and at 15 seconds, 65.4% of sites are error free. This proportion increases to 70.9% by 20 seconds.

General cookies converge similarly, with an error-free median site at 10 second timeouts, 64.3% error-free site at 15 second timeouts, and 70.3% error-free sites at 20 second timeouts.

Takeaway: Across all metrics, the vast majority of privacy-related behavior is captured when using a 15–20 second per-page crawl timeout. Higher timeouts yield diminishing returns in privacy behavior coverage.

4.5 Evaluation Stability

To test temporal robustness, we re-measured a randomly sampled subset of 1K sites five months later using the same configurations and analysis as the initial study. This lets us assess whether crawl behavior and privacy metric observations remain stable over time despite changes in website content and structure.

Crawling Coverage. BFS-based strategies still achieve the highest crawl coverage, finding 50 responsive pages on 72% of sites (79% earlier), followed by breadth-limited BFS at 62% (unchanged) and DFS at 48% (49% earlier).

Crawling Strategies. BFS, stateful execution, and simulated interaction again drive higher metric counts. BFS stateful-with-interaction leads on HTTP-requested domains and Web APIs (tracking and general); for cookies, BFS stateless-with-interaction yields the largest aggregate counts while BFS stateful-with-interaction retains the highest median per-site coverage.

Search Depth. Dynamic termination thresholds for 0% median error remain effectively unchanged: identical for HTTP-requested domains and Web APIs, and at most one page different for cookies.

Crawl Timeout per Page. Three-quarters of sites still show no benefit from timeouts beyond 15 seconds across all privacy metrics.

Takeaway: Across two measurements five months apart, the impact of different crawling configurations remains consistent, supporting the stability and generalizability of our conclusions.

5 Concluding Discussion

Our study presents a detailed evaluation of web crawling strategies for measuring privacy behaviors across the modern web. The results show that crawler configuration significantly affects both the quantity and the variety of privacy metrics collected. In this section, we reflect on key insights from our analysis, summarize best practices, and suggest avenues for future research.

5.1 Web Privacy Crawling Recommendations

First, we discuss our recommendations for future web privacy crawl-based measurements, grounded in our analysis and results. Our findings were consistent across two distinct evaluations done months apart (Section 4.5), providing confidence that our recommendations should apply generally. Our recommendations pertain to the crawler design parameters we evaluated, and not other measurement setup aspects such the browser and framework used, the execution environment, and the region from which a crawl is run. Future work can evaluate how those other aspects influence these recommendations.

Crawling Strategies. Our findings in Sections 4.1 and 4.2 highlight key differences between crawler configurations, particularly around privacy metric quantities and visited sets of site internal pages. In general, we recommend applying BFS for crawl search strategies, as BFS configurations generally collect the highest metric quantities overall, from a broader set of site internal pages, compared to other search strategies. For statefulness and simulated user interaction, we consistently saw benefits from stateful and interactive crawls (with the only exception being cookie-related metrics, where stateless crawls resulted in the most raw number of unique cookies, but mainly due to sites setting freshly-named cookies per page, which likely are not meaningfully different cookies).

Thus overall, we recommend that privacy studies center on BFS stateful crawls simulating user interactions.

Search Depth. Our findings in Section 4.3 showed that different privacy metrics saturate at different crawl search depths, and saturation varies across sites. Web APIs tend to saturate earlier than HTTP-requested domains and cookies, in part due to the finite number of available APIs. Overall, if using a static threshold for deciding when to terminate crawls on each site, a deep crawl is required to ensure high privacy behavior coverage.

Instead, we evaluated a dynamic search termination strategy where a site crawl is terminated if a threshold number of consecutive pages are visited without changes to the observed metric. This approach is adaptive to each site, and we identified it results in high privacy behavior coverage with significantly fewer page crawls on most sites, compared to using a static crawl termination threshold.

Thus, we recommend applying such a dynamic crawl termination strategy when feasible. Our analysis suggests that for our metrics, a dynamic threshold of about 13 pages balances coverage and crawl depth (a threshold of 8 works even better for cookies), with the median site exhibiting no missed privacy behavior (compared to crawling 50 pages).

Note that implementing a dynamic search termination strategy in a live measurement setting is overall straightforward. For each site, the crawler only needs to keep per-site state to track unique privacy metrics and per-page changes, and this state can be discarded once the site's crawl finishes. Even at large scale, the memory overhead is limited to the number of sites being crawled at the same time, making the approach practical in deployment.

Crawl Timeouts per Page. Section 4.4 examined how per-page timeout impacts the coverage of privacy behaviors. The results show that most privacy metrics are captured with 15-20 second timeouts (consistent across search depth). While there is still value in longer timeouts, the gains are increasingly diminishing. Thus, for resource or time-constrained crawls, we recommend a 15 second timeout as a cost-effective balance between crawl time and privacy behavior coverage.

5.2 Future Work

While our work provides an initial systematic investigation of web crawling parameters for web privacy studies, more assessment is still needed. One direction is in expanding the privacy metrics considered. While many privacy measurements are grounded in observed HTTP requests, web APIs, and cookies, this set of metrics is not fully comprehensive. Additional metrics can be evaluated in the future, such as HTTP request volumes, profiling techniques (e.g., canvas fingerprinting), or local data storage.

Another direction is to explore breadth-limited BFS with larger branching factors, as we evaluated a single factor of 3. Exploring more internal links at each level before searching deeper may change how this strategy compares against BFS and DFS.

Also, our analysis looked at raw quantities for privacy metrics, operating under the model that broader observed metrics correlate with more relevant privacy behavior uncovered. However, future work can dive deeper into the actual privacy implications of the behaviors observed, and how these may differ based on crawling configurations. For example, while we identified a crawl configuration that results in higher numbers of unique cookies found on sites, this may not strictly translate to more interesting privacy implications. Thus, for varying privacy objectives, additional investigation is salient for understanding suitable crawling methods.

Furthermore, our study simulated a realistic crawl conducted for a web privacy measurement, but only performed the page search once (although we repeated the crawl on the same pages). Future

work could investigate how different crawl results are if repeating crawls with different page searches, resulting in exploration of a new sets of internal pages each crawl. Our stability analysis in Section 4.5 suggests that the optimal crawl strategy remains consistent, but the actual privacy behaviors observed may differ, which would indicate that repeated crawls provide more coverage of privacy behaviors. Understanding how often to repeat privacy measurement crawls remains an open question for future work.

Ultimately, our study moves toward a grounded understanding of crawling strategies for web privacy measurements, with broader evaluations still needed. We hope our empirically-established guidelines drive more efficient and effective measurements going forward.

References

[1] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. 2014. The web never forgets: Persistent tracking mechanisms in the wild. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[2] Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel. 2013. FPDetective: Dusting the web for fingerprinters. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[3] Syed Suleman Ahmad, Muhammad Daniyal Dar, Muhammad Fareed Zaffar, Narseo Vallina-Rodriguez, and Rishab Nithyanand. 2020. Apophanies or Epiphanies? How Crawlers Impact Our Understanding of the Web. In *Web Conference (WWW)*.

[4] Waqar Aqeel, Balakrishnan Chandrasekaran, Anja Feldmann, and Bruce M Maggs. 2020. On landing and internal web pages: The strange case of jekyll and hyde in web performance measurement. In *ACM Internet Measurement Conference (IMC)*.

[5] Dino Bollinger, Karel Kubicek, Carlos Cotrini, and David Basin. 2022. Automating cookie consent and GDPR violation detection. In *USENIX Security Symposium (USENIX Security)*.

[6] Christian Böttger, Nurullah Demir, Jan Hörnemann, Bhupendra Acharya, Norbert Pohlmann, Thorsten Holz, Matteo Große-Kampmann, and Tobias Urban. 2025. Understanding Regional Filter Lists: Efficacy and Impact. *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2025, 2 (2025), 309–325.

[7] Ahmed Bouhoula, Karel Kubicek, Amit Zac, Carlos Cotrini, and David Basin. 2024. Automated large-scale analysis of cookie notice compliance. In *USENIX Security Symposium (USENIX Security)*.

[8] Stefano Calzavara, Riccardo Focardi, Matteo Maffei, Clara Schneidewind, Marco Squarcina, and Mauro Tempesta. 2018. WPSE: Fortifying web protocols via browser-side security monitoring. In *USENIX Security Symposium (USENIX Security)*.

[9] Darion Cassel, Su-Chin Lin, Alessio Buraggina, William Wang, Andrew Zhang, Lujo Bauer, Hsu-Chun Hsiao, Limin Jia, and Timothy Libert. 2022. OmniCrawl: Comprehensive Measurement of Web Tracking With Real Desktop and Mobile Browsers. 2022, 1 (2022), 227–252. doi:10.2478/popets-2022-0012

[10] Anupam Das, Gunes Acar, Nikita Borisov, and Amogh Pradeep. 2018. The web's sixth sense: A study of scripts accessing smartphone sensors. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[11] Nurullah Demir, Matteo Große-Kampmann, Tobias Urban, Christian Wressnegger, Thorsten Holz, and Norbert Pohlmann. 2022. Reproducibility and replicability of web measurement studies. In *Web Conference (WWW)*.

[12] Nurullah Demir, Jan Hörnemann, Matteo Große-Kampmann, Tobias Urban, Norbert Pohlmann, Thorsten Holz, and Christian Wressnegger. 2023. On the similarity of web measurements under different experimental setups. In *ACM Internet Measurement Conference (IMC)*.

[13] Nurullah Demir, Daniel Theis, Tobias Urban, and Norbert Pohlmann. 2022. *Towards understanding first-party cookie tracking in the field.* arXiv:2202.01498 http://arxiv.org/abs/2202.01498

[14] Nurullah Demir, Tobias Urban, Norbert Pohlmann, and Christian Wressnegger. 2024. A large-scale study of Cookie banner interaction tools and their impact on users' privacy. In *Privacy Enhancing Technologies Symposium (PETS)*.

[15] Clemens Deußer, Steffen Passmann, and Thorsten Strufe. 2020. Browsing unicity: On the limits of anonymizing web tracking data. In *IEEE Symposium on Security and Privacy (S&P)*.

[16] Trinh Viet Doan, Roland van Rijswijk-Deij, Oliver Hohlfeld, and Vaibhav Bajpai. 2022. An Empirical View on Consolidation of the Web. *ACM Trans. Internet Technol.* (2022).

[17] EasyList Project. 2025. EasyPrivacy. https://easylist.to/. Tracking protection filter list. Accessed: 2025.

[18] EasyList Project. 2025. Fanboy's Social, Annoyances, and Notifications Blocking Lists. https://easylist.to/. Accessed: 2025.

[19] Steven Englehardt and Arvind Narayanan. 2016. Online Tracking: A 1-million-site measurement and analysis. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[20] Steven Englehardt, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan Mayer, Arvind Narayanan, and Edward W Felten. 2015. Cookies that give you away: The surveillance implications of web tracking. In *Web Conference (WWW)*.

[21] Imane Fouad, Nataliia Bielova, Arnaud Legout, and Natasa Sarafijanovic-Djukic. 2020. Missed by Filter Lists: Detecting Unknown Third-Party Trackers with Invisible Pixels. In *Privacy Enhancing Technologies Symposium (PETS)*.

[22] Imane Fouad, Cristiana Santos, Arnaud Legout, and Nataliia Bielova. 2022. My Cookie is a phoenix: Detection, measurement, and lawfulness of cookie respawning with browser fingerprinting. In *Privacy Enhancing Technologies Symposium (PETS)*.

[23] Florian Hantke, Peter Snyder, Hamed Haddadi, and Ben Stock. 2025. Web Execution Bundles: Reproducible, Accurate, and Archivable Web Measurements. In *USENIX Security Symposium (USENIX Security)*.

[24] Umar Iqbal, Steven Englehardt, and Zubair Shafiq. 2021. Fingerprinting the fingerprinters: Learning to detect browser fingerprinting behaviors. In *IEEE Symposium on Security and Privacy (S&P)*.

[25] Umar Iqbal, Peter Snyder, Shitong Zhu, Benjamin Livshits, Zhiyun Qian, and Zubair Shafiq. 2020. Adgraph: A graph-based approach to ad and tracker blocking. In *IEEE Symposium on Security and Privacy (S&P)*.

[26] Jordan Jueckstock, Shaown Sarker, Peter Snyder, Aidan Beggs, Panagiotis Papadopoulos, Matteo Varvello, Benjamin Livshits, and Alexandros Kapravelos. 2021. Towards realistic and reproducibleweb crawl measurements. In *Web Conference (WWW)*.

[27] Andrew J Kaizer and Minaxi Gupta. 2016. Characterizing website behaviors across logged-in and not-logged-in users. In *ACM Internet Measurement Conference (IMC)*.

[28] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. 2016. Beauty and the Beast: Diverting modern web browsers to build unique browser fingerprints. In *IEEE Symposium on Security and Privacy (S&P)*.

[29] Zengrui Liu, Jimmy Dani, Yinzhi Cao, Shujiang Wu, and Nitesh Saxena. 2025. The First Early Evidence of the Use of Browser Fingerprinting for Online Tracking. In *Web Conference (WWW)*.

[30] Shaoor Munir, Patrick Lee, Umar Iqbal, Zubair Shafiq, and Sandra Siby. 2024. PURL: Safe and effective sanitization of link decoration. In *USENIX Security Symposium (USENIX Security)*.

[31] Shaoor Munir, Sandra Siby, Umar Iqbal, Steven Englehardt, Zubair Shafiq, and Carmela Troncoso. 2023. Cookiegraph: Understanding and detecting first-party tracking cookies. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[32] Meenatchi Sundaram Muthu Selva Annamalai, Emiliano De Cristofaro, and Igor Bilogrevic. 2025. Beyond the Crawl: Unmasking Browser Fingerprinting in Real User Interactions. In *Web Conference (WWW)*.

[33] ChangSeok Oh, Chris Kanich, Damon McCoy, and Paul Pearce. 2022. Cart-ology: Intercepting targeted advertising via ad network identity entanglement. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[34] OpenWPM. 2020. OpenWPM JS Instrumentation. https://github.com/openwpm/OpenWPM/blob/master/docs/Configuration.md#js_instrument.

[35] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. 2016. Website fingerprinting at Internet scale. In *Annual Network and Distributed System Security Symposium (NDSS)*.

[36] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2018. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Annual Network and Distributed System Security Symposium (NDSS)*.

[37] Tanya Prasad, Rut Vora, Soo Yee Lim, Nguyen Phong Hoang, and Thomas Pasquier. 2026. RegTrack: Uncovering Global Disparities in Third-party Advertising and Tracking. In *Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb)*. doi:10.14722/madweb.2026.23010

[38] Jannis Rautenstrauch, Metodi Mitkov, Thomas Helbrecht, Lorenz Hetterich, and Ben Stock. 2024. To auth or not to auth? A comparative analysis of the pre-and post-login security landscape. In *IEEE Symposium on Security and Privacy (S&P)*.

[39] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. 2012. Detecting and defending against third-party tracking on the web. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*.

[40] Kimberly Ruth, Deepak Kumar, Brandon Wang, Luke Valenta, and Zakir Durumeric. 2022. Toppling top lists: Evaluating the accuracy of popular website lists. In *ACM Internet Measurement Conference (IMC)*.

[41] Iskander Sanchez-Rola, Matteo Dell'Amico, Davide Balzarotti, Pierre-Antoine Vervier, and Leyla Bilge. 2021. Journey to the center of the cookie ecosystem: Unraveling actors' roles and relationships. In *IEEE Symposium on Security and Privacy (S&P)*.

[42] Asuman Senol, Alisha Ukani, Dylan Cutler, and Igor Bilogrevic. 2024. The double edged sword: Identifying authentication pages and their fingerprinting behavior. In *Web Conference (WWW)*.
[43] Sachin Kumar Singh, Robert Ricci, and Alexander Gamero-Garrido. 2025. Where in the World Are My Trackers? Mapping Web Tracking Flow Across Diverse Geographic Regions. In *ACM Internet Measurement Conference (IMC)*. doi:10.1145/3730567.3764427
[44] Peter Snyder, Antoine Vastel, and Ben Livshits. 2020. Who filters the filters: Understanding the growth, usefulness and efficiency of crowdsourced ad blocking. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 4, 2 (2020), 1–24.
[45] Aleksei Stafeev and Giancarlo Pellegrino. 2024. SoK: State of the krawlers – evaluating the effectiveness of crawling algorithms for web security measurements. In *USENIX Security Symposium (USENIX Security)*.
[46] Tobias Urban, Martin Degeling, Thorsten Holz, and Norbert Pohlmann. 2020. Beyond the front page: Measuring third party dynamics in the field. In *Web Conference (WWW)*.
[47] Tobias Urban, Dennis Tatang, Martin Degeling, Thorsten Holz, and Norbert Pohlmann. 2020. The unwanted sharing economy: An analysis of cookie syncing and user transparency under GDPR. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS)*.
[48] Antoine Vastel, Pierre Laperdrix, Walter Rudametkin, and Romain Rouvoy. 2018. FP-STALKER: Tracking browser fingerprint evolutions. In *IEEE Symposium on Security and Privacy (S&P)*.
[49] Yash Vekaria, Vibhor Agarwal, Pushkal Agarwal, Sangeeta Mahapatra, Sakthi Balan Muthiah, Nishanth Sastry, and Nicolas Kourtellis. 2021. Differential tracking across topical webpages of indian news media. In *ACM Web Science Conference (WebSci)*.
[50] Rahulkrishna Yandrapally, Andrea Stocco, and Ali Mesbah. 2020. Near-duplicate detection in web app model inference. In *International Conference on Software Engineering (ICSE)*.

A Ethics

Our primary ethical concern relates to the crawling of websites as part of our measurement study. To minimize impact on websites, we applied strict rate-limiting, ensuring at most one page is crawled per site per minute (usually much slower). We additionally capped requests at 100 pages per site, with up to 50 active pages collected, thus limiting the total amount of resources accessed per website. Crawling this amount at our limited rate should not tax even a small website, and our measured population of sites are among the most popular sites. Furthermore, the metrics collected by the browsers as part of the study are publicly visible data, and we do not impact real website users through our study nor do we collect personal information.

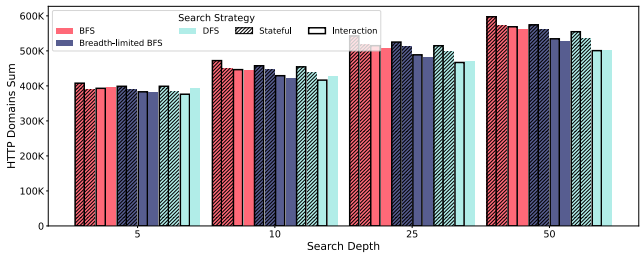


Figure 13: Sum of the number of unique HTTP-requested domains observed per site, aggregated across all sites, when searching to varying depths.

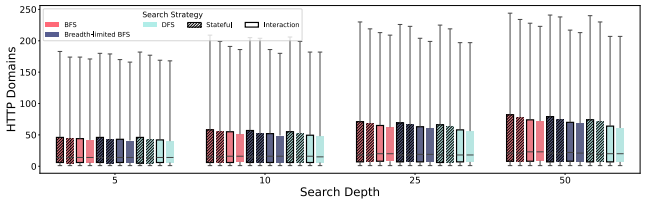


Figure 14: Number of unique HTTP-requested domains observed per site, when searching to varying depths. The top whisker represents the 95th percentiles.

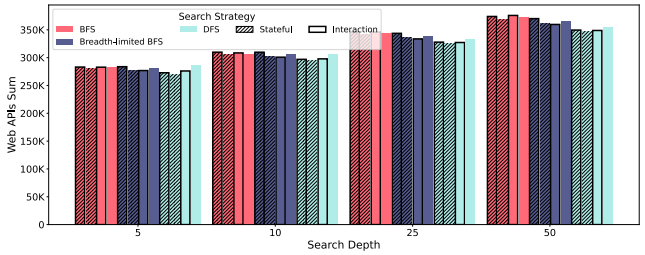


Figure 15: Sum of the number of unique Web APIs observed per site, aggregated across all sites, when searching to varying depths.

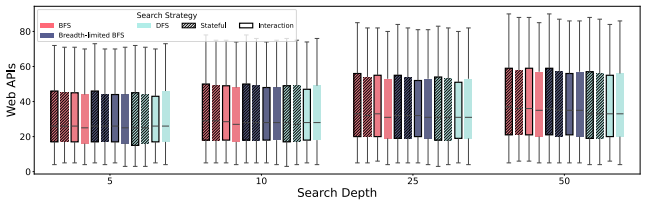


Figure 16: Number of unique Web APIs observed per site, when searching to varying depths. The top whisker represents the 95th percentiles.

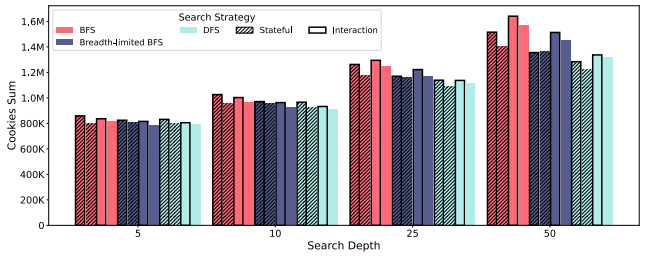


Figure 17: Sum of the number of unique cookies observed per site, aggregated across all sites, when searching to varying depths.

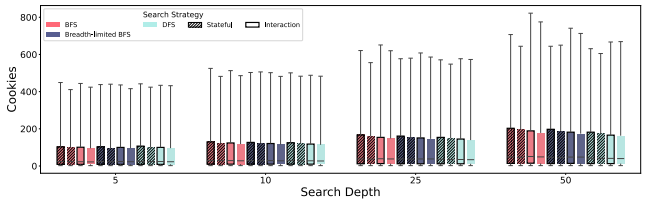


Figure 18: Number of unique cookies observed per site, when searching to varying depths. The top whisker represents the 95th percentiles.

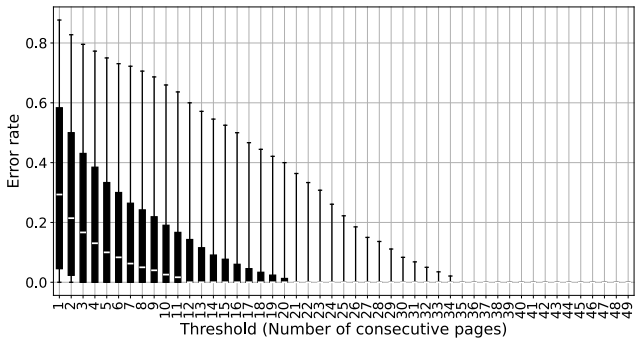


Figure 19: Boxplot of the per-site Fingerprinting API error rates, across sites and when terminating using various dynamic thresholds, where the error rate is defined as the fraction of Fingerprinting APIs missed with dynamically terminating, relative to the total APIs found by a full crawl (up to depth 50). The top whisker represents the 95th percentiles.

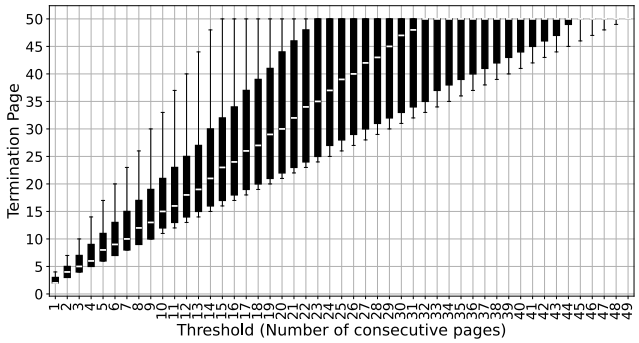


Figure 20: Boxplot of when crawls terminate on sites (the termination page) under the dynamic termination strategy, considering different thresholds for consecutive pages with unchanged Fingerprinting APIs count. The whiskers represent the 5th and 95th percentiles.

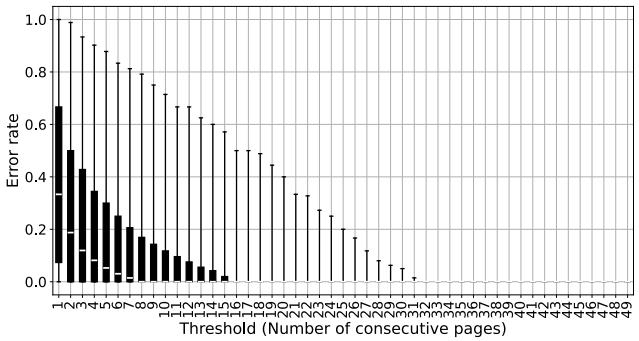


Figure 21: Boxplot of the per-site tracking cookie error rates, across sites and when terminating using various dynamic thresholds, where the error rate is defined as the fraction of tracking cookies missed with dynamically terminating, relative to the total cookies found by a full crawl (up to depth 50). The top whisker represents the 95th percentiles.

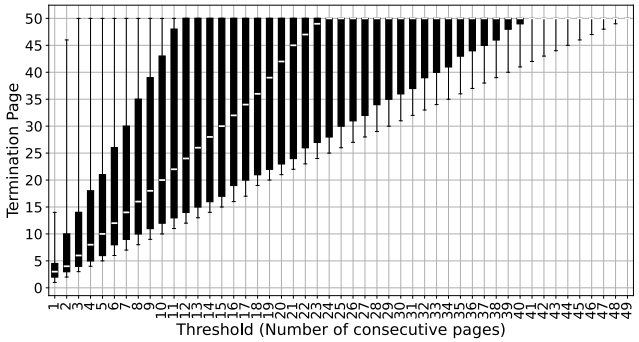


Figure 22: Boxplot of when crawls terminate on sites (the termination page) under the dynamic termination strategy, considering different thresholds for consecutive pages with unchanged tracking cookies count. The whiskers represent the 5th and 95th percentiles.